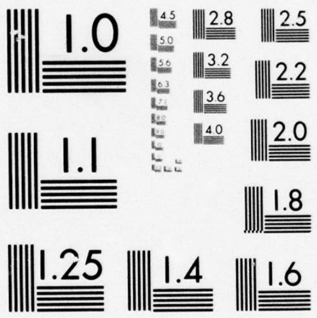
AD-A040 592 TEXAS UNIV AT AUSTIN CENTER FOR CYBERNETIC STUDIES F/G 9/2
IMPROVED COMPUTER-BASED PLANNING TECHNIQUES.(U)
FEB 77 F GLOVER, J HULTZ, D KLINGMAN N00014-75-C-0616
UNCLASSIFIED CCS-283 NL

1 OF 1
AD A040 592

END
DATE FILMED
7-77



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12 NW

CENTER FOR CYBERNETIC STUDIES

The University of Texas
Austin, Texas 78712

Research Report CCS 283

IMPROVED COMPUTER-BASED PLANNING TECHNIQUES

by

Fred Glover*
John Hultz**
Darwin Klingman***

February 1977

*Professor of Management Science, University of Colorado, Boulder, CO 80302

**Senior Systems Analyst, Analysis, Research and Computation, Inc., Box 4067, Austin, TX 78767

***Professor of Operations Research and Computer Sciences and Director of Computer Science Research Center for Cybernetic Studies, University of Texas, BEB 608, Austin, TX 78712

Invited paper at the SHARE XLVIII Conference in Houston, March 6-11, 1977.

This research was partly supported by ONR Contract N00014-76-C-0383 with Decision Analysis and Research Institute and by Project NR047-021, ONR Contracts N00014-75-C-0616 and N00014-75-C-0569 with the Center for Cybernetic Studies, The University of Texas. Reproduction in whole or in part is permitted for any purpose of the United States Government.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
Business-Economics Building, 203E
The University of Texas
Austin, Texas 78712
(512) 471-1821

ABSTRACT

Management Science has responded recently to the needs of practitioners by contributing two new technologies. These are *network computer implementation technology* and *NETFORM (network formulation) technology*. In this paper we show how these new technologies have been used to model and solve real-world problems. In addition, we attempt to give the practitioner insights into how these important policy evaluation tools may be applied to his unique management problems.

OVERVIEW

The growth of the computer industry has had a profound influence on many areas, affecting none more dramatically than the area of *management science*. A virtual explosion of new knowledge about ways to solve optimization problems in industry and government has occurred since World War II, much of it intimately dependent upon the capability provided by the computer to record and manipulate extremely large amounts of data. Without this capability, many of the tools of management science would be mere theoretical niceties.

One of the most important areas brought into being by the advent of the computer is that of *computer-based* planning models. The techniques for building, solving, refining, and analyzing such planning models have undergone a steady evolutionary development as computer hardware has changed. This evolution has recently spawned two new and important technologies, *network computer implementation technology* and *NETFORM (network formulation) technology*.

Computer implementation technology has received a marked impetus from recent research on new solution algorithms and implementation techniques for solving network problems [2, 3, 5, 11, 13, 14, 17, 19, 28]. The fruits of this research have dramatically reduced the cost of solving the broad class of problems in the linear and mixed integer network domain. This cost reduction, significantly, has been entirely above and beyond any reductions afforded by changes in computer hardware or compilers. For example, the cost of solving network problems with 2400 equations and 500,000 arcs on an IBM 360/65 has been reduced from $10,000 in 1968 to $300 in 1976. In addition these advances have stimulated the development of new modeling techniques for handling a multitude of problems that arise in applications of scheduling, routing, resource allocation, production, inventory management, facilities location, distribution planning, and other areas.

These new modeling techniques [12, 15, 16, 18] are mathematically and symbolically linked to network and augmented network structures and constitute the central focus of *NETFORM (network formulation) technology*. A major attribute of the NETFORM technology is that it allows users to conceptualize formulations of their problems graphically. The pictorial aspect of this technology has proven to be extremely valuable in both communicating and refining problem inter-relationships without the use of mathematics and computer jargon. Thus it protects the non-technical person against technical legerdemain and exaggerated claims of model "realism." Another powerful attribute of this technology is that it often yields a model which can be solved as a sequence of linear network problems.

One purpose of this paper is to briefly describe the network computer implementation and NETFORM innovations and to demonstrate their power when used in *concert* to model and solve real-world applications. We argue on the basis of *practical* experience that these advances overcome many of the conceptual design and problem solving difficulties of previous approaches to system optimization. Moreover, they provide the type of technologies required of truly useful decision planning tools. Our implementations of these tools in industry and government have led us to the conclusion that the weakest link in developing effective computer-based planning models is no longer in the computer and management science tools themselves, but rather in the way these new tools are applied.

The ultimate test and worth of these tools, however, depends on their use by practitioners. Thus the more important purpose of this paper is to convey a sense of what is now possible and offer some insights that will make it easier for practitioners to take advantage of these developments.

HISTORICAL HIGHLIGHTS

From the inception of mathematical programming in the late forties and early fifties, networks have constituted one of the most significant classes of practical application. Beginning in the late fifties, however, there was a hiatus of about ten years during which practitioners and theoreticians alike focussed attention on other models and solution algorithms which bore no visible connection to networks. The measure of merit often seemed to be the level of mathematical abstraction that could be achieved, as though by ever more rarified generality it might be possible to encompass the entire realm of optimization in a single theoretical framework.

Two difficulties ensued. The first was that many of the models and methods were notably opaque, and not readily communicated. The second difficulty was that, by glossing over details at lower levels of abstraction, there were few solid results that led to efficient treatment of problems from the practical domain. This led companies to surrender to expedient but less than fully satisfactory solution processes of an ad hoc nature. Many (though not all) of the efforts classed as simulation fall conspicuously in this category.

Such simulation and related "quasi-optimizing" approaches, while extensively used today, do not allow valid inferences about the quality of solutions obtained. As input parameters are changed, the user cannot adequately determine whether the differences of the solutions are due to different parameter values or to the inability of the solution process to sift through the multitude of feasible solutions to find good ones. This makes it almost impossible to validate such models. In addition, these approaches often require a lot of computer time. A novel but telling example is that it requires 2 cpu minutes to simulate each single minute of a computer's operation.

Another pitfall of the shift to highly abstract models, made apparent by 20-20 hindsight, was that the process of obtaining solutions to such models became a numbers-in/numbers-out game which told *what* but not *why*. That is, at the important strategic level of decision making, the solution to a model can only provide supporting guidelines and insights into the decision to be made. The solution *cannot* fully characterize the nature or scope of the decision in itself. Increasing the complexity or level of abstraction of a model does not alter this fact. In addition, the common practice of solving a model once, for a single set of assumed conditions or relationships, does not provide adequate insight into how a system responds to environmental changes such as price or demand fluctuations.

To garner significant insights requires that a model be iteratively solved using a systematically developed sequence of supply/demand/policy scenarios. Practical (i.e., *useful*) understanding rests on a laborious synthesis of the solutions for different scenarios, interpreted and moderated through the experience of the analyst. This was scarcely facilitated by the abstract model approaches prominent throughout most of the sixties. Practitioners were quick to observe that solutions to complex algebraic models are extremely difficult to collate and synthesize.

In response to this situation, special schematic modeling procedures were developed both for representing input problem data and for exhibiting solutions. For example, Shell Oil Company developed a graphical problem generation system for mathematical optimization problems involving refinery operations. The authors, working for different industrial firms, devised a variety of highly simple but effective pencil-and-paper and computer procedures for viewing and examining solutions.

As the use of these procedures became more widespread, an important observation resulted: Many problems that were previously expressed only as complex algebraic models can be given a pictorial, network-related formulation. Moreover, such a formulation does not sacrifice essential rigor, but is mathematically equivalent to the algebraic statement of the problem. This observation, derived from applications and not theory, marks the founding point of the NETFORM technology.

In the subsequent sections we present examples of how algebraic models can be viewed graphically, describe several real-world applications that have profited by the use of NETFORM techniques, compare and contrast network computer codes with commercial linear programming codes, and briefly discuss how network techniques may be integrated into linear programming codes to produce the next generation of linear programming systems.

PART I MODELS AND MODELING TECHNIQUES

PURE NETWORK MODELS

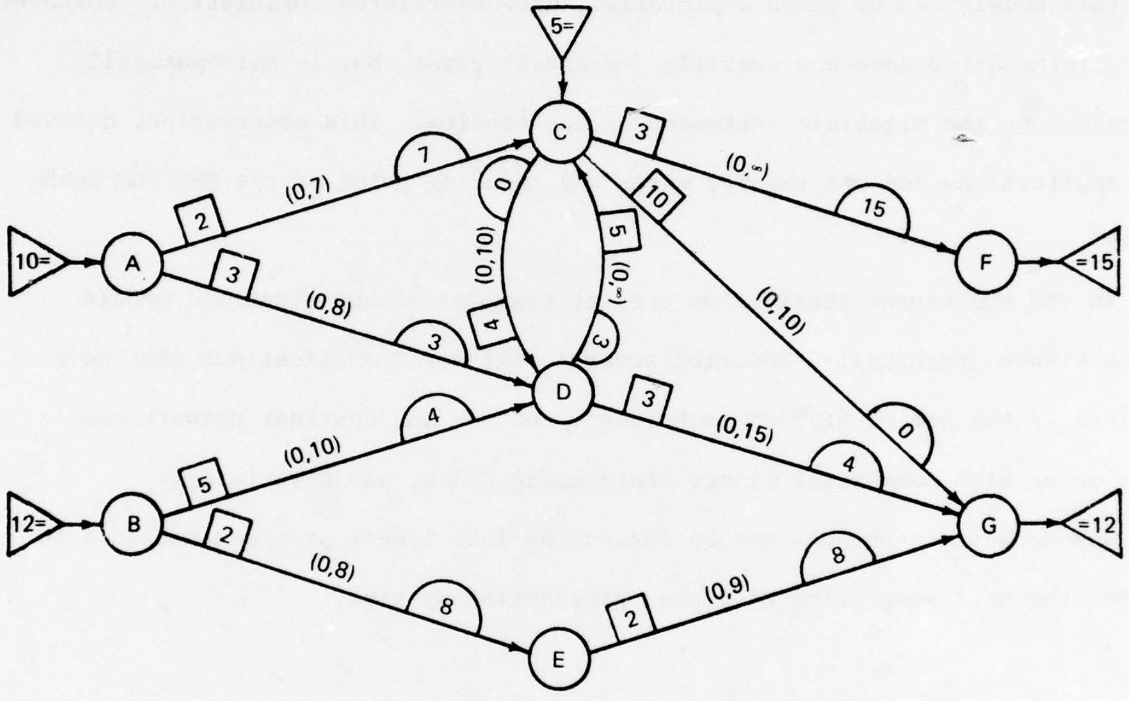
Pure network problems actually embody a group of distinct model types. This group includes shortest path, assignment, transportation, and transshipment problems. Any of these network problems can be characterized by a coefficient matrix which has at most one +1 and one -1 entry in each column. For the sake of unification and brevity we will focus attention on the most general of these model types.

Transshipment Problem

The *transshipment model* appears in numerous applications, either directly, or as a subproblem. An illustration of this model for a cash flow problem is depicted in Figure 1.

Figure 1

Capacitated Transshipment Cash Flow Problem



The *arrows* shown in Figure 1 are called *arcs* and the *circles* are called *nodes*. In this cash flow network, the nodes may be thought of as corresponding to subsidiaries of a central company that operates in different locations. The *supplies* and *demands*--which are shown in the triangles leading into a node for a supply and out of a node for a demand--may be thought of as representing excess or deficit cash positions. Thus, nodes A, B, and C have excess funds, nodes D and E have no funds, and nodes F and G have deficit funds.

The arcs in this illustration indicate the ways to transfer cash from one subsidiary to another. For instance, the arc from node A to node C indicates that it is possible to transfer funds from subsidiary A to subsidiary C. The absence of an arc indicates that it is not possible to transfer funds directly between the corresponding pair of subsidiaries (though it may be possible to transfer funds indirectly by means of a sequence of arcs through intermediate subsidiaries). Each arc has a lower bound, upper bound, and cost. The lower and upper bounds appear within the parentheses and the cost within the rectangle on an arc. For example, Figure 1 indicates that the arc from node A to node C has a lower bound of 0, an upper bound of 7, and a cost of 2.

The objective in the transshipment problem is to determine how much to ship along each arc within the limits stipulated by the bounds in order to satisfy all supplies and demands and to minimize total cost. By satisfying supplies and demands we mean that the total flow into the node minus the total flow out must equal its demand, and the total flow out of the node minus the total flow in must equal its supply. For all other nodes the flow into the node must equal the flow out. The numbers in the semi-circles on the arcs in Figure 1 illustrate a solution which satisfies these *node equations* and the bound requirements.

Mathematical Representation of the Transshipment Problem

It is quite important to understand how a transshipment problem may be stated mathematically in order to appreciate the connections between graphical and algebraic structures. We will build on this fundamental understanding in later sections of the paper.

To state a network problem algebraically, define a variable for each arc. For example, let X_{ij} denote the flow on the arc from node i to node j and c_{ij} denote the unit cost on this arc. Henceforth, an arc will be denoted as an ordered pair (i,j) where the first component specifies the *from node* and the other component the *to node.*

Next create the objective function for the problem as an expression involving the costs and variables. For the problem in Figure 1, the objective function would be:

$$2X_{AC}+3X_{AD}+5X_{BD}+2X_{BE}+4X_{CD}+3X_{CF}+10X_{CG}+5X_{DC}+3X_{DG}+2X_{EG}$$

Upon identifying the objective function, create a constraint for each node which expresses the restriction on flow into and out of the node. To do this, it is convenient to view a supply as an inflow and a demand as an outflow. Then the requirement at each node can be expressed as Total Inflow = Total Outflow, or in particular, Total Inflow - Total Outflow = 0.

The customary transposition of any constant term of this equation to the right hand side causes supplies to be denoted as negative quantities and demands to be denoted as positive quantities. To see this, consider constructing the "Total Inflow - Total Outflow =0" equation for node C in Figure 1. The total flow into node C consists of the 5 units of supply and $(X_{AC} + X_{DC})$. The total flow out of node C is $X_{CF} + X_{CG} + X_{CD}$. Thus we have $5 + X_{AC} + X_{DC} - (X_{CF} + X_{CG} + X_{CD}) = 0$, or after transposing the constant term, $X_{AC} + X_{DC} - X_{CF} - X_{CG} - X_{CD} = -5$. Thus the supply of 5 becomes expressed as a negative quantity because of its movement to the right hand side of the equality sign. On the other hand, a demand, (which is included in the outflow that is subtracted in the "Inflow-Outflow" equation), appears as a positive quantity when moved to the right hand side. This also discloses, incidently, that a supply may be viewed as a negative demand, and vice versa.

The entire algebraic statement of the capacitated transshipment problem shown in Figure 1 is as follows:

Minimize: $2X_{AC} + 3X_{AD} + 5X_{BD} + 2X_{BE} + 4X_{CD} + 3X_{CF} + 10X_{CG} + 5X_{DC} + 3X_{DG} + 2X_{EG}$

Subject to:

$$\begin{array}{cccccccccccc}
-X_{AC} & -X_{AD} & & & & & & & & = & -10 \\
& & -X_{BD} & -X_{BE} & & & & & & = & -12 \\
X_{AC} & & & & -X_{CF} & -X_{CG} & +X_{DC} & & & = & -\ 5 \\
& X_{AD} & +X_{BD} & & & & -X_{DC} & -X_{DG} & & = & 0 \\
& & & X_{BE} & & & & & -X_{EG} & = & 0 \\
& & & & X_{CF} & & & & & = & 15 \\
& & & & & X_{CG} & & +X_{DG} & +X_{EG} & = & 12
\end{array}$$

$0 \le X_{AC} \le 7;\ 0 \le X_{AD} \le 8;\ 0 \le X_{BD} \le 10;\ 0 \le X_{BE} \le 8;\ 0 \le X_{CD} \le 10;$

$0 \le X_{CD};\ 0 \le X_{CF};\ 0 \le X_{CG} \le 10;\ 0 \le X_{DC} \le 15;\ 0 \le X_{EG} \le 9$

It is important to observe that each X_{ij} appears in *exactly two* node equations, i.e., the equation for node i and the equation for node j. Further, since X_{ij} contributes to the outflow of the node i equation and to the inflow of the node j equation, it appears with a coefficient of -1 in the former and with a coefficient of +1 in the latter. Thus, each column of the coefficient matrix has one -1 and one +1 entry. If the restrictions at the nodes are stated as inequalities rather than equations (i.e., Total Inflow - Total Outflow ≤ 0 or Total Inflow - Total Outflow ≥ 0), then slack or surplus variables are added to convert these constraints to equations and it is admissible for columns to contain a single non-zero entry. Inequality restrictions at nodes generally arise by stipulating that supplies or demands must be "at least" or "at most" a certain amount.

Note that, by reversing the above steps, it is possible to represent any LP problem whose coefficient matrix has at most one +1 and at most one -1 entry per column as a transshipment problem. That is, to construct a graph that corresponds to such a problem, create a *node* for each *constraint* and an *arc* for each *variable*, affixing supplies, demands and bounds in the manner indicated.

APPLICATIONS OF PURE NETWORK PROBLEMS

Pure network problems provide models for numerous mathematical optimization problems and major compenents of many additional problems. Inventory maintenance problems [9, 29, 31], for example, typically exhibit an underlying network structure. A cousin of the inventory maintenance problem is the so-called PERT/CPM problem, which seeks the best way to sequence a complex set of interdependent activities. The PERT/CPM framework, which constitutes one of the simplest network model forms, has been used in a variety of practical applications (including construction of the Polaris submarine) and has been reported to save substantial dollar costs and greatly speed the completion of complex projects.

Other types of problems involving the effective management of resources also frequently exhibit network structures. Such problems are becoming increasingly important in government and industry. Direct network formulations of water resource management problems, for example, are finding use in a number of states. In these, canals, river reaches, and pipelines take the role of arcs, while reservoirs and pumping stations take the role of nodes. Planning over time frequently looms as a major consideration in these applications.

The Texas Water Development Board and the government of Poland introduce "what if" analyses into water resource management by using a succession of simulations of alternative "supply configurations," and solve the resulting network for each simulation run. (The step of finding the optimum solution to each network problem is used to determine the best response to meet demands for water use, given a particular supply configuration. This use of simulation, in which parameters are varied to achieve "what if" analyses by means of vigorous solution techniques, is to be contrasted with the commonly encountered "quasi-optimization" use of simulation discussed earlier.) To analyze the full range of relevant configurations, roughly five hundred such runs must be made each month. The feasibility and cost-effectiveness of such runs of course owes heavily to the efficiency of solving the underlying networks.

The problem of determining flows and heads in a general pipeline system (such as in municipal water systems) with reservoirs, pumps, gate and check valves, given fixed inputs and withdrawals has been recently shown in [25] to be equivalent to a convex transshipment problem under the assumption of convex head losses. Such problems are easily solved as ordinary transshipment problems using a piecewise linear approximation of the convex function. Since the convexity requirements are usually satisfied for real pipe networks, this is an example of another class of real-world problems which can now be handled by network procedures with far greater effectiveness than by the procedures applied to these problems in the past.

Another important instance of the use of network models occurs in manpower promotion and assignment problems. AT&T has developed such models in order to guarantee acceptable hiring and promotion policies in accordance with HEW rules and regulations. As the numerical illustration of the preceding section suggests, a number of cash management problems have also recently been modeled as transshipment problems. These models include sources of funds in addition to cash (such as maturing accounts and notes receivable, sales of securities, borrowing, etc.)

and uses of funds other than a single "investment." The generalized network model to be discussed subsequently makes it possible to further incorporate discount, interest, and other financial considerations directly into the model.

Many nonlinear problems involve network subproblems. One of the most basic and prevalent forms of nonlinear problems is the fixed-charge network problem, whose major offshoots include the extremely important genre known as "location" problems. The nonlinear element of a fixed-charge network is the fixed-charge arc which has the following special property: whenever the arc is "used" (i.e., permitted to transmit flow), a charge is incurred that is independent of the amount of flow across the arc. Fixed-charge networks have been used to model problems of plant and warehouse location, equipment purchasing and leasing, personnel hiring and offshore oil drilling platform location, among others.

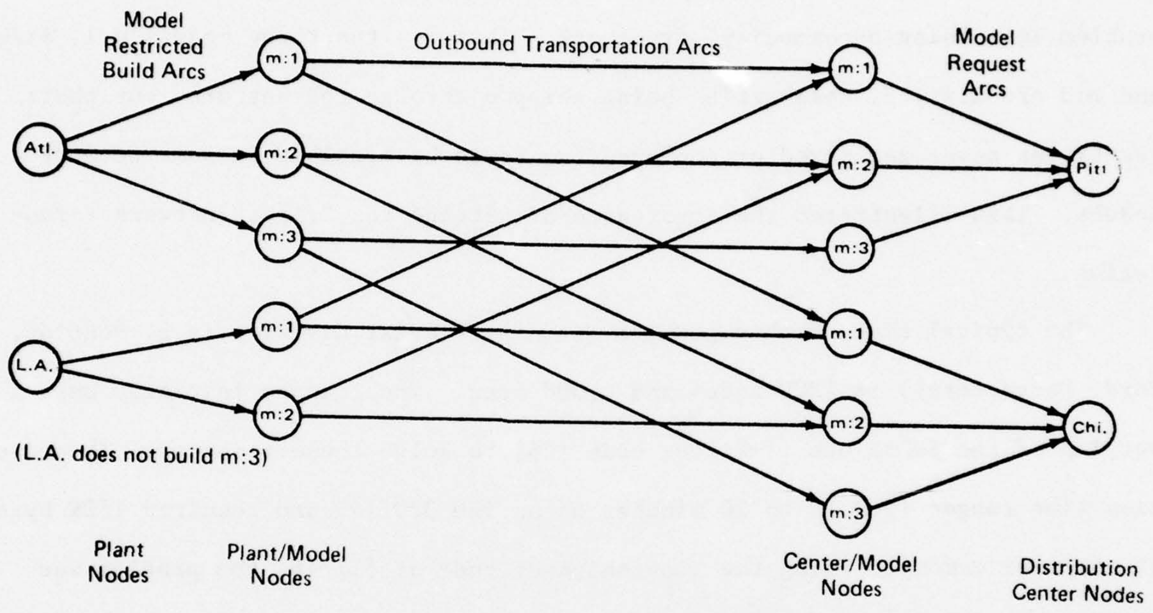
To provide a fuller appreciation of the ingredients of such models, we will now discuss in detail an example from an important practical application.

Production Planning and Distribution Application

A major U. S. car manufacturer has developed and implemented a transshipment model for production planning and distribution decisions. This model is noteworthy for demonstrating the value of networks in interactive decision making. Figure 2 illustrates this application.

The problem, in simplified form, is to determine the number of cars of each of three models (m:1, m:2, and m:3) to produce at the Atlanta and Los Angeles plants (represented by the "Atl." and "L.A." nodes), and then to determine how many of each of these car models to ship from each plant to the distribution centers in Pittsburgh and Chicago (represented by the "Pitt." and "Chi." nodes). The objective is to identify a production-distribution plan that minimizes total cost.

Figure 2
Production Planning and Distribution



Model Restricted Build Arcs
Outbound Transportation Arcs
Model Request Arcs
Atl.
L.A.
m:1
m:2
m:3
Pitt
Chi.
(L.A. does not build m:3)
Plant Nodes
Plant/Model Nodes
Center/Model Nodes
Distribution Center Nodes

Bounded supplies are associated with the Atl. and L.A. nodes, indicating the least and most that can be produced at these plants. In addition, upper and lower bounds are placed on the various arcs emanating from these two nodes to control the minimum and maximum number of each particular car model that can be produced at these plants. Similar bounds (capacity restrictions) can also be placed on other arcs. For instance, if there is a limit on the number of m:1 cars that can be shipped from Atlanta to Pittsburgh, then this appears as an upper bound restriction on the "top center" arc in the network. Finally, the number of each particular model required at Pittsburgh and Chicago is handled by placing bounds on the "far right" arcs. For example, if exactly 4,000 m:3-type cars are required in Chicago, then 4,000 becomes both the lower and upper bound on the m:3-Chi. arc.

An interesting feature of this model is not only that it coordinates the production and distribution decision, but that it handles a multi-commodity problem in a "single-commodity" framework. That is, the three models m:1, m:2, and m:3 are distinct commodities being shipped through the network, but their identities never get mixed or confused, as could be possible in some network models. This illustrates the importance of getting the "right" network formulation.

The typical size of this problem for a particular division (e.g. Pontiac, Ford, Dodge, etc.) is 1200 nodes and 4,000 arcs. The company initially used a version of the SHARE out-of-kilter code [26] to solve these problems. The solution time ranged from 10 to 20 minutes on an IBM 370/145 and required 150K bytes of computer memory. Using the transshipment code of [1, 14] the problem was solved in less than 20 seconds on the company's IBM 370/145, using only 80K bytes of computer memory, thus making it possible to solve such problems in an on-line computer mode. In fact, due to the nature of the decision making environment of this application, the company has developed an on-line real time production planning and distribution system which is linked to a graphics display terminal and an English language input processor. This system is currently being used at several administrative levels within the corporation hierarchy for planning purposes.

Today, by using the interactive on-line network system and utilizing visual displays, plant executives are able to discuss their goals and assumptions in a very short time. Answers to questions of the form "What if we do this?" are quickly obtained and evaluated. In fact, the executives typically are able to evaluate 150-200 production plans each quarter with the aid of the network system.

GENERALIZED NETWORK

The generalized network (GN) problem represents a class of LP problems that heretofore has received only a small portion of the attention it has deserved. Recently, however, with the identification of many new generalized network applications and with the emergence of computer codes able to solve these problems efficiently, generalized networks are coming to be appreciated as rivaling or even surpassing pure networks in their practical significance. Generalized networks include pure networks as a special case. The GN problem, by allowing non-zero column entries other than ± 1, is actually the broadest classification of linear network-related problems. Practical settings in which such GN problems arise include resource allocation, production, distribution, scheduling, capital budgeting, as well as other problem types to be discussed.

As previously noted, the most effective procedures for modeling and communicating pure network problems are based on viewing these problems as directed graphs. A generalized network problem can also be represented as a directed graph by means of appropriate conventions. In particular, the coefficient matrix can be transformed (if the variables are bounded) so that at least one entry in any column with two non-zero entries is -1. In this way, a directed arc is created that leads from the node associated with the -1 to the node associated with the other non-zero entry. (If both entries are -1, the arc may be directed either way.) A single non-zero entry in a column is represented by an arc that touches only one node (which is both its starting and ending point.)

There is an important distinction between arcs in pure network problems and arcs in GN problems. An arc of a generalized network has a *multiplier*. This multiplier is the non-zero coefficient associated with the node at the head of the arc (i.e., the terminal node of the directed arc). In pure networks, this multiplier is always +1.

These ideas are illustrated by the following GN problem.

Minimize: $1X_{12} + 5X_{13} + 3X_{23} + 1X_{24} - 4X_{32} - 9X_{34}$

Subject to:

$$\begin{array}{rrrrrrcr}
-1X_{12} & -1X_{13} & & & & & = & -5 \\
2X_{12} & & -1X_{23} & -1X_{24} & +1/3X_{32} & & = & 0 \\
 & 1/2X_{13} & +1X_{23} & & -1X_{32} & -1X_{34} & = & 0 \\
 & & & -1/5X_{24} & & +3X_{34} & = & 10
\end{array}$$

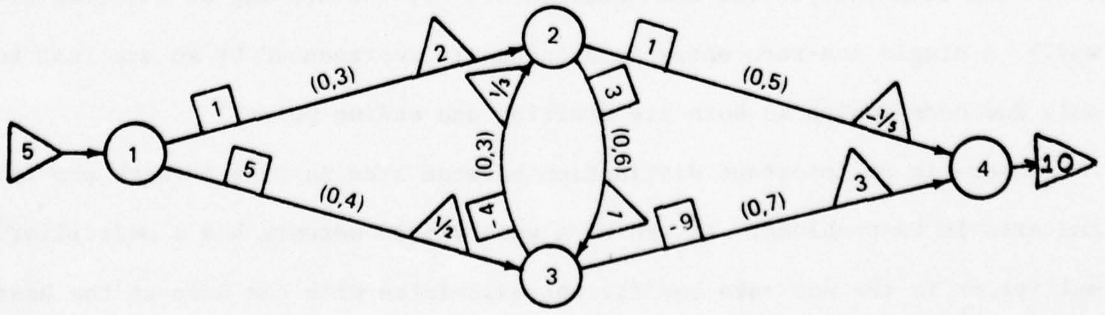
$$0 \le X_{12} \le 3,\ 0 \le X_{13} \le 4,\ 0 \le X_{23} \le 6$$

$$0 \le X_{24} \le 5,\ 0 \le X_{32} \le 3,\ 0 \le X_{34} \le 7$$

The associated network is shown in Figure 3. As with pure network problems, each row of the coefficient matrix is associated with a node and each column with an arc.

Figure 3

Generalized Network



That is, a node corresponds to a problem equation and an arc corresponds to a problem variable. Consequently, each arc has a cost, lower bound, and upper bound. Costs are shown in Figure 3 within the squares and bounds are shown within parentheses. Arc multipliers are shown within triangles.

The multiplier of a generalized network problem acts upon the flow across an arc so that the amount of flow starting out on the arc will not necessarily be the amount arriving at the opposite end. Specifically, the flow entering the arc is multiplied by the value of the multiplier to produce the quantity of flow leaving the arc. For example, if 2 units start on the arc from node 1 to node 2 in Figure 3, the multiplier of 2 will cause 4 units to arrive at node 2. Likewise, 10 units starting on the arc from node 2 to node 4 will result in 2 units arriving at node [illegible] since the multiplier in this case is 1/5. It is important to keep in mind that *the arc's cost, lower bound, and upper bound refer only to the units of flow entering the arc.*

APPLICATIONS OF GENERALIZED NETWORKS

Generalized networks can successfully model many problems that have no pure network equivalent. This is made possible by two useful interpretations of arc multipliers. First, multipliers can be viewed as modifying the amount of flow of some particular item. By means of flow modification, generalized networks can model such situations as evaporation, seepage, deterioration, breeding, interest rates, sewage treatment, purification processes with varying efficiencies, machine efficiency, and structural strength design. However, it is also possible to interpret the multiplication process as transforming one type of item into another. This interpretation provides a way to model such processes as manufacturing, production, fuel to energy conversion, blending, crew scheduling, manpower to job requirements, and currency exchanges. The following applications (see [6, 12, 15, 16]) lend insight into the possible uses of generalized networks.

A complete water distribution system with losses has been modeled by Bhaumik [4] as a generalized network problem. This model is primarily concerned with the movement of water through canals to various reservoirs. However, the model must also consider the retention of water over several time periods. The multipliers in this case represent the loss effect due to both evaporation and seepage.

Turner and Gilliam [10] have proposed a file reduction model which has the form of a generalized transportation model with a single extra constraint. This model is designed to facilitate the reduction of extremely large microdata files to smaller, statistically representative files. The objective is to minimize the amount of information lost in the reduction process. The arcs represent paths from the original records to the reduced records. A non-zero flow on an arc implies that the originating record is to be represented by the terminal record. The multipliers on the arcs are used to insure that the reduced file is truly representative of all of the original records.

Kim [23] has utilized generalized networks to represent copper refining processes. The electrolytic refining procedure is modeled by a large d-c electrical network. The arcs are current paths with the multipliers representing the appropriate resistances. In this way, Kim analyzes the effect of short circuits in the refining process.

Charnes and Cooper [6] identify applications of generalized networks for both plastic-limit analysis and warehouse funds-flow models. In plastic-limit analysis, the network is generated by forming the equations for horizontal and vertical equilibrium and by employing a coupling technique. The warehouse funds-flow model is actually a multi-time period model. The arcs are used to represent sales, production, and the inventory holding of both products and cash. The multipliers are introduced to facilitate the conversions between cash and products.

A cash management problem for a multi-national firm has been modeled as a generalized network by Crum [7]. This model incorporates transfer pricing, receivables and payables, collections, dividend payments, interest payments, royalties, and management fees. The arcs represent possible cash flow patterns and the multipliers represent costs, savings, liquidity changes, and exchange rates.

Other applications of generalized networks include machine loading problems [6, 8, 31], blending problems [6, 31], the caterer problem [8, 31], and scheduling problems such as production and distribution problems, crew scheduling, aircraft scheduling, and manpower training [6, 8, 31].

INTEGER GENERALIZED NETWORKS

An especially significant realm of application, only recently discovered, extends the use of generalized networks by requiring that flows on particular generalized arcs must occur in integer (whole number) amounts. Introducing the integer requirement into the GN problem enables it to model an unexpected diversity of additional applications, including problems such as scheduling variable length television commercials into time slots, assigning jobs to computers in computer networks, scheduling payments on accounts where contractual agreements specify "lump sum" payments, and designing communication networks with capacity constraints.

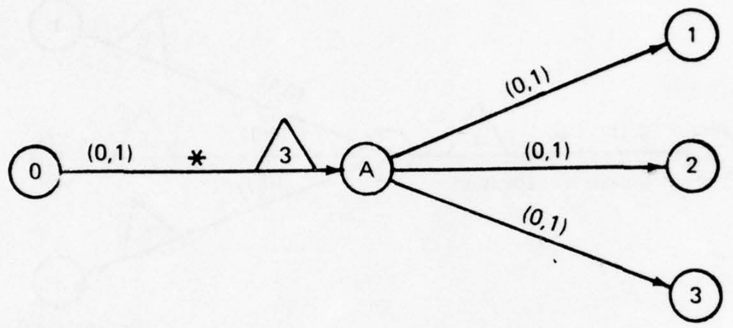
In fact, the use of integer requirements in GN problems has been embodied in model techniques that are among the mainstays of NETFORM technology. The power of these NETFORM techniques is illustrated by the fact that they enable any 0-1 LP problem to be modeled as an integer GN problem [16, 18]. These techniques can also accommodate mixed integer 0-1 LP problems where the continuous part of the problem is a transportation, transshipment or generalized network problem itself. An illustration in [30] shows how contemporary financial capital allocation models can be modeled as integer GN problems. Important real-world applications with such a "mixed" structure also include a variety of plant location models, energy models, and physical distribution models.

In general, the NETFORM approach also uses model techniques based on requirements that flows on certain arcs must equal their upper or lower bounds, or that at least (or at most) a specified number of arcs from particular sets must have 0 flows and other similar "logical" requirements. (Many of these, by appropriate transformation, can also be converted into model structures involving integer constrained GN problems.) These NETFORM representations consisting of networks with easily specified "side conditions," are able to rigorously express all problem elements that would ordinarily require expression in an abstract algebraic form (as by customary mathematical programming formulation techniques). Consequently, they effectively *replace* the obscure and unilluminating algebraic representation by an equivalent, but much easier to understand, pictorial representation. We will show how this comes about by several examples, beginning with a fundamental technique that reappears in various guises in a remarkable variety of practical settings. The later examples also demonstrate that the underlying network-related structures of the NETFORM approach can often be exploited by special solution methods that are far more efficient than the methods previously developed for the algebraic representations.

Figure 4 illustrates a useful modeling device based on integer constrained generalized arcs commonly employed in the NETFORM approach. The bounds, costs, and multipliers are depicted by the same conventions employed before. In addition, the *asterisk* on the arc from node 0 to node A indicates that its flow must be an *integer*.

Figure 4

Generalized Network with Interger Flow Restrictions

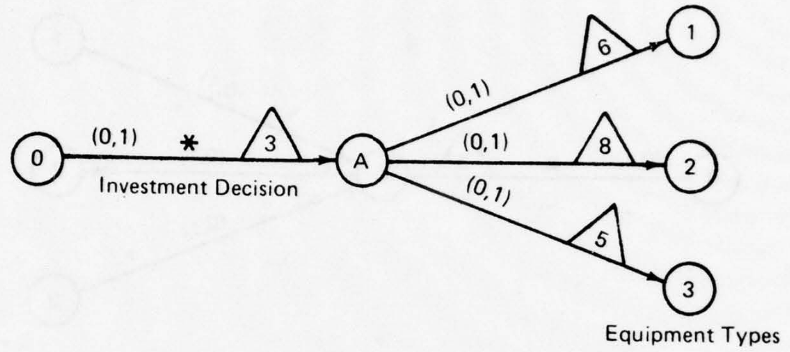


If the flow is 0, then 3 · 0 = 0 and no flow gets transmitted to node A. But if the flow is 1, then 3 units are transmitted to node A. Further, because of the upper bounds of 1 on each of the three arcs leaving node A, the only possible way to distribute the 3 units flowing into node A is to send exactly one unit to each of the nodes 1, 2, and 3. Thus, by giving all arcs bounds of 0 and 1, and introducing a generalized arc, the following effect has been achieved: *when the flow on the arc from node 0 to node A is 0, the flow on each of the three arcs out of node A is 0; when the flow on the arc from node 0 to A is 1, the flow on each of the three arcs out of node A is 1.*

The extension of this device to handle an even more useful set of conditions is illustrated in Figure 5. This figure is the same as before except that multipliers have now been added to the three arcs leaving node A. For concreteness, we may suppose this diagram represents an investment decision: to invest in project A (if the flow on the arc from 0 to A is 1) or not to invest in project A (if the flow on the arc from 0 to A is 0). Then the nodes 1, 2, and 3 identify

Figure 5

An Equipment Investment



different components of this investment project. In this example, the investment project is set in an equipment purchase context, and nodes 1, 2, and 3 represent different equipment types. In other contexts, these nodes might represent different types of aircraft in a fleet, different parcels of land in a real estate venture, different types of stock in a portfolio, etc.

The multipliers on the arcs into nodes 1, 2, and 3 represent the quantities of each component of project A (each type of equipment) that would be acquired if in fact the decision is made to invest in that project. By the conventions and flow relationships previously described, the diagram of Figure 5 transforms the investment into its components in precisely the manner desired; i.e., in particular a flow of 1 on the arc from node 0 to node A (representing the decision to invest) translates into six units of Equipment 1 at node 1, eight units of Equipment 2 at node 2, and five units of Equipment 3 at node 3. The combination of arc multipliers and the 0-1 integer restriction gives rise to what generally is called an *integer network* or a *0-1 generalized network*. The kinds of uses to which this NETFORM modeling tool can be put are demonstrated more fully by the following real-world applications.

Air Force Course Scheduling

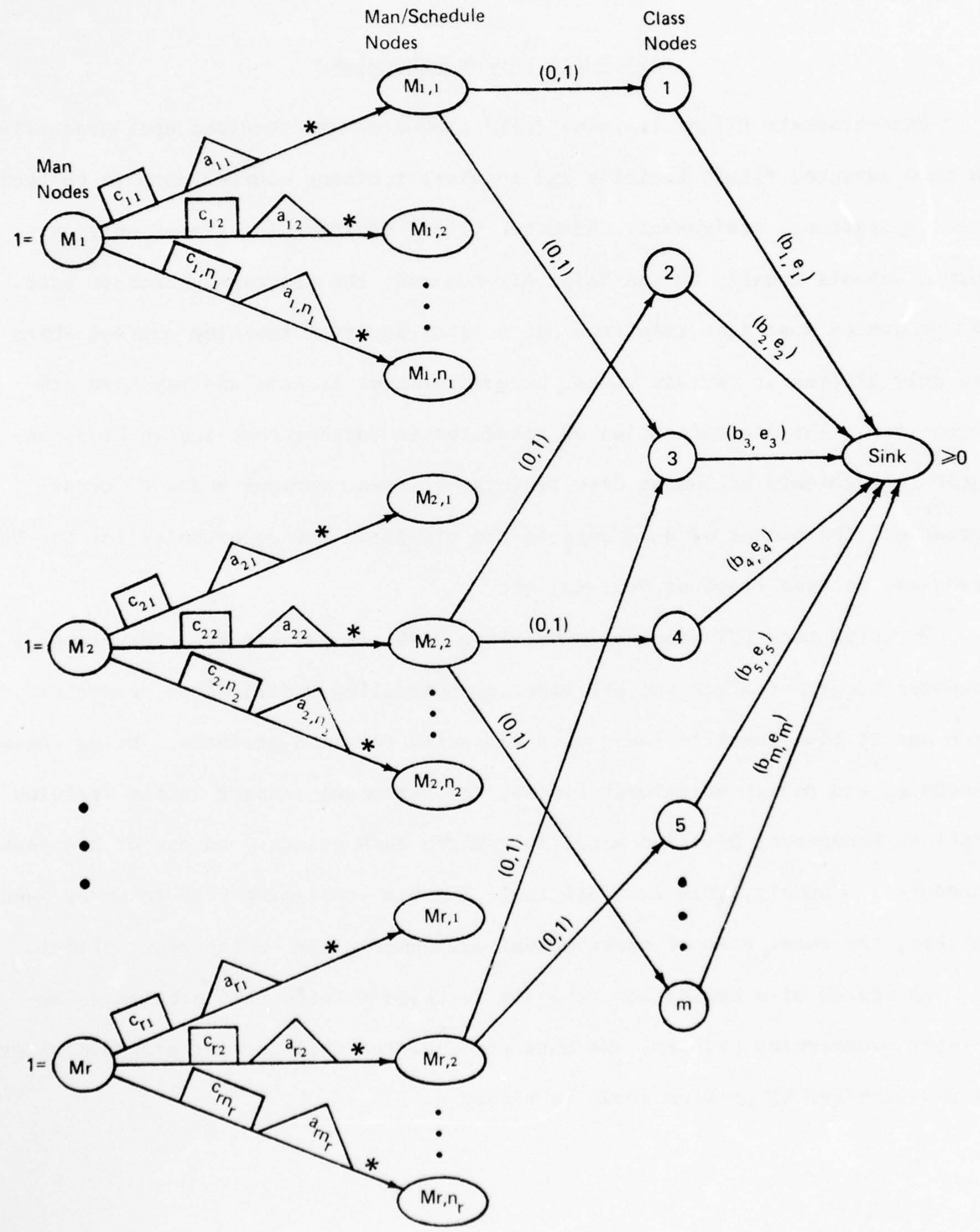
Undergraduate Flight Training (UFT) graduates are required upon graduation to take advanced flight training and survival training courses enroute to their first operational assignment. Advanced flight training is offered only in formal schools usually by the Major Air Command, the principal aircraft user. UFT graduates must also take from one to four survival training courses which are only offered at certain times, have enrollment limits, and may have prerequisites. The identification of schedules is further complicated by attendance requirements at Combat Crew Training courses, various modes of transportation, the number of dead days in the pipeline, the opportunity for the UFT graduates to take leave as desired, etc.

To solve this UFT graduate scheduling problem, the Air Force developed a computer program (called the UFT Pipeline Scheduling Model) which generates from one to five feasible least cost schedules for each graduate. Using these schedules and course enrollment limits, the personnel manager in the Training Pipeline Management Division *manually* assigns each graduate to one of his feasible schedules. Clearly, this is a difficult and time-consuming task to do by hand; further, the total cost of these manual assignments may be far from optimal.

In search of a better way, the Air Force formulated this problem as an integer programming problem. We have reformulated this integer programming problem as the 0-1 GN problem shown in Figure 6.

Figure 6

UFT NETFORM FORMULATION



In this diagram, the node Mi represents the i^{th} man and has a supply of exactly 1. Each man node is connected by arcs to its set of man/schedule nodes. Each connecting man/schedule arc has a multiplier a_{ij} equal to the number of classes in the schedule and a cost c_{ij} equal to the cost of assigning man i to his j^{th} schedule. The asterisk again indicates that flow must be integer-valued. The arcs emanating from a man/schedule node in Figure 6 lead to the individual classes making up the schedule. Each of these arcs has an upper bound of one. Thus, if a particular schedule is "selected," then every class in the schedule is also automatically selected. The objective is to pick a schedule for each man that will minimize the value of the assignments on the overall program, subject to the upper and lower attendance limits for each class, expressed as bounds on the arcs from class nodes to the sink nodes of Figure 6. All arc costs, except for those attached to the man/schedule arcs, are thus equal to 0. The essence of this problem, which we have been able to state rigorously in a single paragraph by means of the accompanying NETFORM diagram, is far more difficult to ferret out and communicate from the algebraic mathematical programming formulation of the integer programming problem, as may be imagined.

Typically the UFT problem involves 120 men, 200 classes, and 460 schedules, giving rise to a 0-1 formulation with 520 constraints and 460 0-1 variables. The 0-1 formulation involves 460 0-1 variables, 2,200 continuous variables and 780 nodes. This represents a fair increase in size as an LP problem, but provides a relatively small GN problem. Indeed, after initial testing, the Air Force judged the original 0-1 problem too difficult to solve optimally within reasonable time limits. By contrast, we were able to solve the NETFORM representation of this problem quite easily using a specialized branch and bound procedure with GN subproblems. The optimal solution was often found and verified after only 30 seconds and in some cases only required a total solution time of 10 seconds on a CDC 6600.

Refueling Nuclear Reactors

Another problem whose solution has been improved by the use of the NETFORM concept is a mixed integer programming problem for determining the minimum cost refueling schedule for nuclear reactors. This problem was initially modeled by Kazmersky [22] as a mixed integer programming problem with no apparent connection to networks. However, after working closely with Dr. Kazmersky, we discovered a way to express the problem by a NETFORM representation that was not only equivalent to the original formulation but that also succeeded in reducing the size of the problem. The transformation of the original problem to a 0-1 GN problem will not be shown because the mathematics is somewhat intricate and the original formulation by itself consumes more than twenty pages of [22]. However, the simplifications produced by the NETFORM representation for conveying the nature of the problem are comparable to those provided by the other example cited here. The full model can be described in a handful of pages without the use of complex mathematical notation by means of this approach.

In addition, making use of the 0-1 GN formulation, we were able to develop a branch and bound solution procedure which solved GN subproblems, yielding significant gains over previous solution efforts. Using this computer system, four versions of this problem were solved using data supplied by the TVA. The first three versions, while requiring half an hour to two hours to solve on an IBM 370/168 using the MPSX solution system, were easily solved in 10 to 20 minutes using the 0-1 GN formulation and the specialized solution approach. The fourth version was by far the most difficult, involving 173 constraints, 126 zero-one variables, and 511 continuous variables. The original mixed integer formulation was run for eight hours on an IBM 370/168 using MPSX and then taken off the machine to avoid further computer run costs. At the end of this time the best (minimum cost) solution obtained had an objective function value of \$136,173,440. With a time limit of 30 minutes imposed on the 0-1 GN solution effort, a solution was obtained that had an objective function value of \$125,174,727, which constitutes more than a \$10,000,000 improvement. Consequently, this application again shows that the use of the NETFORM approach can provide improved solutions for problems too complex to be solved optimally (within practical time limits) by standard approaches.

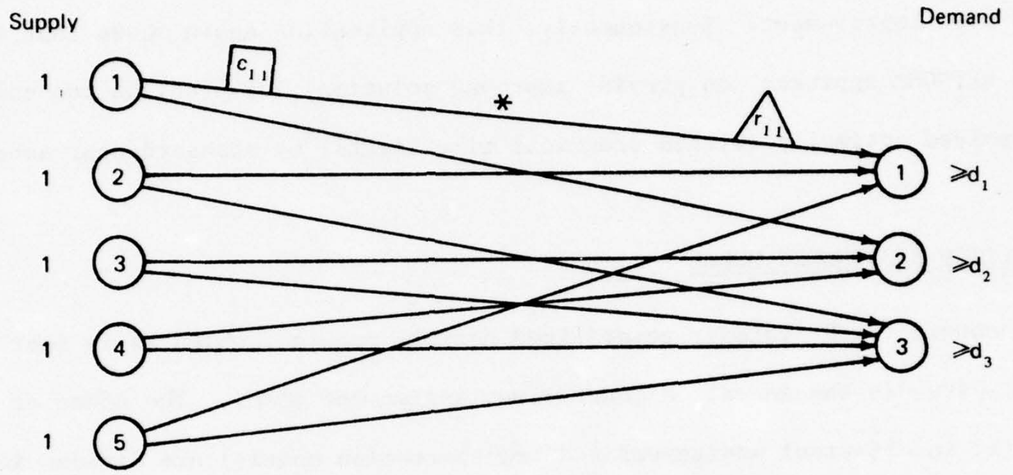
GENERALIZED ASSIGNMENT MODEL

A cousin to the integer generalized network problem, which is in fact a special case, is the so-called *generalized assignment model*. The nodes of this model (as in classical assignment and transportation models) are divided into two sets. The nodes of the first set are called *origin nodes* and only have arcs leaving them while the nodes of the second set are called *destination nodes* and only have arcs entering them, as depicted in Figure 7. Each origin node has a supply of exactly one and each destination node, subscripted by the index i, has a demand of at least (or at most) d_i. Additionally, each arc has a cost, a multiplier, and an integer (0-1) restriction. For example, c_{11}, r_{11}, and the asterisk on the arc between origin node 1 and destination node 1 of Figure 7 indicate the cost, multiplier, and integer requirement on the arc.

The name "generalized assignment model" is due to an instance of the model in which it is desired to assign personnel (at the origin nodes) to jobs (at the destination nodes). Each job has a demand that requires at least a certain number of "standard man-hours" (i.e., man-hours of an individual at a standard skill level) to be devoted to that job during a specified interval (weekly, monthly, etc.). The multiplier r_{ij} on an arc represents the number of standard man-hours a given person can contribute to a given job during the interval. Note that an

Figure 7

Generalized Assignment Problem



individual will have different multiplier values on different arcs as the result of possessing a skill that exceeds or falls below the standard. An additional requirement is that each person can only be assigned to one job (integer requirement). In contrast to the classical assignment problem [6, 8, 31], however, a job may have several persons assigned to it in order to satisfy its demand.

In another application, the origin nodes are visualized as checking accounts and the destination nodes as days of the month. The demands are "at most" requirements ($\leq d_i$) and represent daily auditing capacity. Each account must be assigned to a day of the month and the multipliers represent the auditing effort required for the account.

A further application of this model involves the assignment of ships to shipyards for overhaul. The origin nodes in Figure 7 can represent the ships and the destination nodes shipyards. Like the last application, the demands are "at most" requirements and represent shipyard capacity in days. The multiplier represents the number of days required for the overhaul at the shipyard. The "cost coefficients" could be a weighted combination of attributes such as transportation costs, overhaul costs, desirability of assigning the ship to this yard, etc. The objective would then be to minimize total "cost."

A major real-world application of this problem that demonstrates the nature and practical value of its model structure more fully will now be described.

Optimal Lot-Sizing and Machine Loading for Multiple Products

The generalized assignment model described in this section is currently being used by a major manufacturing firm for large-scale task allocation. The problem involves the determination of lot-sizes for each of n products and the assignment of production to m machines in such a way that combined set-up, production, and holding cost per unit time is minimized. The principal characteristics of the problem are:

1. The planning horizon is a single period, t weeks in length.
2. The products are designed to meet different needs and cannot be substituted for one another. Production of each product is a single-stage process.
3. Lot-sizes are selected from a predetermined finite set of ℓ possible lot-sizes.
4. All lots of any single product must be produced on the same machine.
5. The machines work in parallel. They are similar in function, but they may differ in their rate and cost of operation. Some machines may be capable of producing several (or all) of the products while others may be more specialized.
6. The production capacities of all machines over the planning horizon are known constants. Each machine can produce only one product at a time.
7. Demand for each product is assumed to occur continuously at a known constant rate.

In this application we will begin by describing the mathematical programming formulation in algebraic form, in order to demonstrate the translation *from* such a formulation *to* a NETFORM representation (as opposed to developing a NETFORM representation directly from the initial problem conditions). (This indirect formulation route was in fact the one taken historically in this application, since the algebraic form of the problem turns out to be quite simple, though the direct approach is usually easier and preferable.)

For the algebraic representation, binary valued decision variable x_{ijk} is introduced which is defined to be 1 if product j is produced on machine i in the k^{th} possible lot-size and 0 otherwise. The combined set-up, production, and holding cost (per unit time) incurred when product j is produced on machine i in the k^{th} possible lot-size is denoted by c_{ijk}. Similarly, r_{ijk} denoted the capacity required on machine i to produce product j in the k^{th} possible lot-size. Finally, b_i denotes the aggregate production capacity of machine i over the t week planning horizon.

The problem is to determine values for the variables that

$$\text{Minimize:} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{\ell} c_{ijk} x_{ijk}$$

$$\text{Subject to:} \quad \sum_{i=1}^{m} \sum_{k=1}^{\ell} x_{ijk} = 1 \text{ for } j = 1,\dots,n \tag{1}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{\ell} r_{ijk} x_{ijk} \le b_i \text{ for } i = 1,\dots,m \tag{2}$$

$$x_{ijk} = 0 \text{ or } 1 \text{ for } i = 1,\dots,m; \quad j = 1,\dots,n; \; k = 1,\dots,\ell \tag{3}$$

Analysis discloses that constraints (1) and (3) together insure that one and only one machine and lot-size combination is selected for each product. Constraint (2) insures that each machine is assigned production tasks commensurate with its capacity.

This formulation is designed only to load the machines and does not schedule the work on each machine. In fact, the lot-sizes selected by the model may generate scheduling conflicts on any given machine. The reason for allowing such potential conflicts is that managers primarily use the model for capacity planning and prefer to retain the option of scheduling on the basis of what is "hot." Thereby, they retain the prerogative of determining the precise sequence and timing for implementing the candidate assignment over the horizon, in accordance with the objectives of this application. This provides flexibility to make adjustments to special conditions and changed demands, while simultaneously aiding planning functions (such as evaluating the possible use of overtime shifts in periods when the candidate assignments tax weekly production capacities). For this type of flexibility and responsiveness to the needs of management, and to further support the analyses based on alternative assumptions of demands and capacities, it is especially important to be able to solve the model quickly for different (or recently updated) sets of data. Thus, the success of the application depends in large measure on the ability to solve the problem efficiently.

The firm in which this application arises initially tried to solve the problem using the efficient 0-1 code RIP 30-C developed by Geoffrion, Graves, and McBride at UCLA. This proved to be unsuccessful for two reasons: (1) the large array requirements of RIP 30-C made it impossible to accommodate large problems; and (2) the method required excessive computation times even to solve problems with no more than 50 variables.

Consequently, it was apparent that an alternative solution approach was needed. The first step of our effort to identify such an approach was to characterize the network related structure of the problem, which in this case turns out to be a generalized assignment structure, as already intimated.

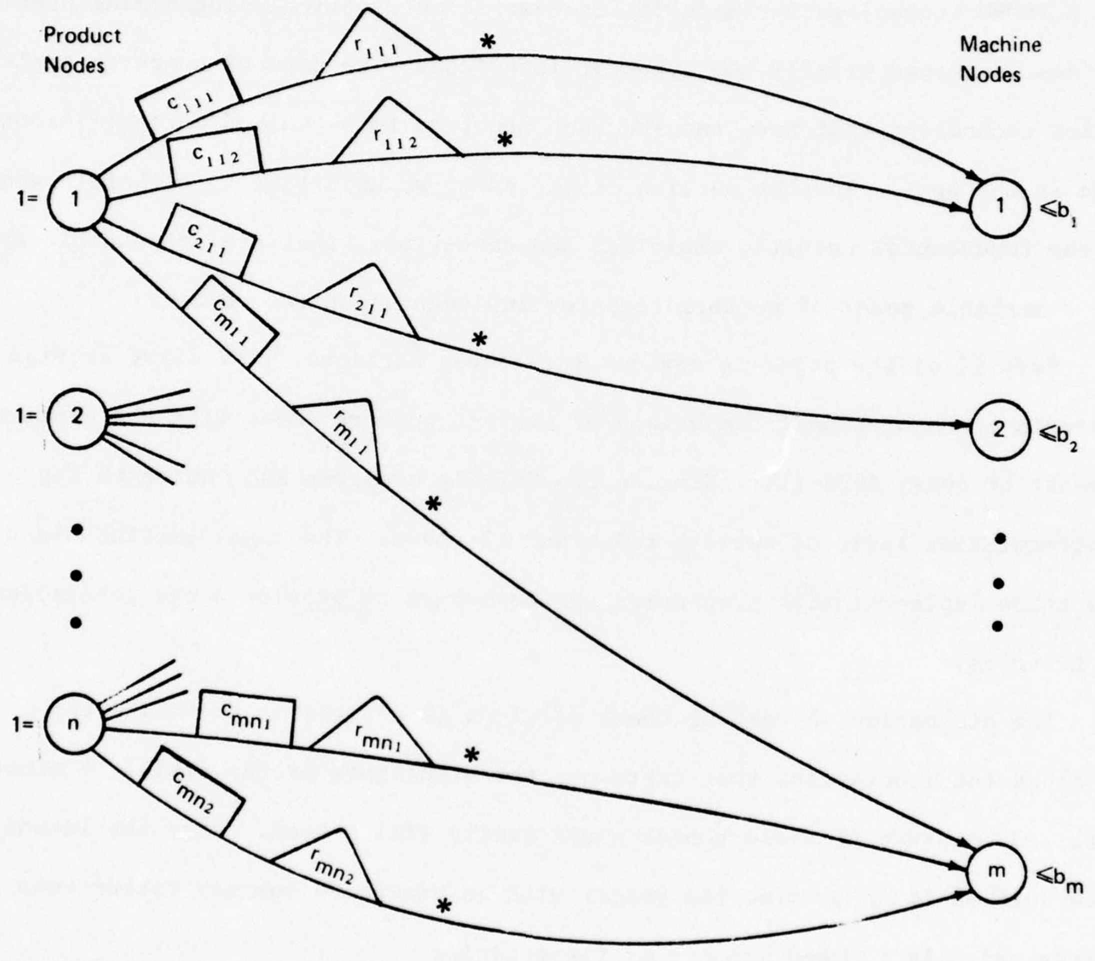
Although this is not immediately apparent from the algebraic statement, if equations (1) and (2) are written out (replacing (1) by its negative), the coefficient matrix has only two non-zero entries per column, which are -1 and r_{ijk}. Thus this problem can be represented graphically in the usual fashion by letting a node represent each equation and an arc each variable. Figure 8 illustrates the resulting graph. The two sets of nodes for this problem consist of a product node set and a machine node set. The arcs joining the nodes in these sets correspond to the variables x_{ijk}; the cost and multiplier of each of these arcs correspond to the cost and resource consumption of the associated variable.

This graphical representation as a *generalized assignment* problem led immediately to identifying an appropriate solution method. In particular, extremely effective techniques for solving such problems have been developed by Ross and Soland [27] and imbedded in a computer code called Big-A.

A comparison of the Big-A code with the RIP 30-C code for this problem shows that the Big-A code is from 300 to 1000 times faster. In addition, the Big-A code readily handles problems of up to 4000 variables within available computer memory. Thus, the firm now uses the graphical formulation coupled with the Big-A code to solve the problem. This approach has made it possible to solve problems with 106 machines, 182 products, 4 lot-size options per machine/product combination and 3772 zero-one variables in .64 seconds on a CDC 6600 and 10 seconds on an IBM 370/145.

Figure 8

NETFORM REPRESENTATION



PART II ALGORITHMS AND IMPLEMENTATIONS

Part I of this paper has provided some insights into the nature and value of the NETFORM technology for modeling real-world mathematical programming problems, and has indicated briefly the types of interfaces with network computer implementation technology that have enabled such problems to be solved more efficiently than in the past. In this portion of the paper we undertake to elaborate some of the fundamental results, empirical and conceptual, that are responsible for the remarkable power of network computer implementation technology.

Part II of the paper is divided into three sections. The first section presents a computational comparison of current network codes with the state-of-the-art LP code, APEX-III. Section two briefly compares and contrasts the implementation logic of network codes and LP codes. The final section discusses how these implementation procedures can be merged to provide a new generation of LP codes.

The discussion in each of these sections is devoted to sketching the concepts and innovations that represent the highlights of the field. A minutely detailed coverage of these topics would easily fill a book. Thus the intent of each section is to provide the reader with an executive summary rather than a microscopically focused account of these topics.

COMPARISON RESULTS

The impressive computational statistics presented in Part I for solving problems with network-related structure rest largely on recent advances in solution methods for pure and generalized assignment, transportation and transshipment problems. In this section we take a closer look at the form of these advances. In particular, we provide fuller insight into the practical significance of these advances by presenting the results of empirical tests of the new methods against a leading example of the state-of-the-art in solution systems that does not incorporate the network technology. To do this, we report computational comparisons of the new network codes against APEX-III on a wide array of network problems of varying structures.

These results are not biased by variations in computer hardware: all problems were solved on the same machine. Further, solution runs were executed during time periods in which comparable demands were made upon the computer. Even with these safeguards, minor differences between two solution times should be statistically ignored and the focus should be on order of magnitude differences. For this reason, the times reported are for large problems so that timing variations become less significant.

Table I contains solution times on 15 network problems using APEX-III on a CYBER-74. The first set of problems consists of assignment problems and the reported network solution times were obtained using the AP-AB code of [3]. The solution times indicate that the AP-AB code is roughly 200 times faster than APEX-III on assignment problems.

The network code times reported on the transportation and transshipment problems were obtained using the ARC-II code of [2]. Again the network solution times are substantially superior (on the order of 130 times faster than APEX-III). The fourth set of solution times are for generalized network problems. The network code times refer to the NETG code [12].

The relative superiority of network code times to APEX-III is smaller for generalized networks than for pure networks. The code NETG is on the order of 50 times faster than APEX-III on generalized networks; nevertheless, this superiority is dramatic, especially in terms of computer costs for solving such

TABLE I

(times are in billing units)

PROBLEM TYPE	no. of equations	no. of variables	APEX III solution times	APEX III cost	Network Code solution times	Network Code cost
Assignment	400	1500	231.85	$ 41.73	1.16	$ .21
	400	2250	336.37	$ 60.55	1.34	$ .24
Transportation	200	1300	105.68	$ 19.02	.94	$ .17
	200	1500	124.53	$ 22.42	1.07	$ .19
	200	2000	164.94	$ 29.69	1.21	$ .22
Transshipment	400	1306	174.83	$ 31.47	1.51	$ .27
	1000	2900	833.63	$150.05	5.28	$ .95
Generalized networks	250	4000	453.02	$ 81.54	16.65	$ 3.00
	250	4000	742.61	$133.67	14.74	$ 2.65
	500	5000	1044.34*	$187.98	22.55	$ 4.06
	1000	6000	1633.64*	$294.06	50.22	$ 9.04
Singularly constrained generalized networks	200	2000	205.87	$ 37.06	16.10	$ 2.90
	200	1000	130.18	$ 23.43	11.38	$ 2.05
	500	4000	943.25	$169.79	32.72	$ 5.89
	1000	6000	1875.55*	$337.60	83.13	$14.96

* Not optimal after 10,000 iterations.

problems. For example, using the NETG code, one could solve a GN problem every week for a year, and incur roughly the same cost as by solving the problem only once using APEX-III.

The final set of test results are for LP problems composed of a GN problem augmented by an arbitrary linear constraint. This problem-type is called the *singularly constrained generalized* network problem. The network code times refer to the NETSG [20] code and are approximately 25 times faster than APEX-III. The results even for this class of constrained generalized networks are *more than an order of magnitude* faster than available with the advanced LP system.

In addition to improving solution speed, the network processing techniques have the noteworthy advantage of requiring less computer memory to solve a problem. This allows larger problems to be solved without resorting to external storage devices, which can incur significant cost increases due to lengthened computer run times. Further, the reduced memory requirements enable many computer-based decision systems that would otherwise be excluded from this option to be used in an interactive real-time processing environment.

Yet another important advantage of the network codes is their portability. All of these codes are written in standard FORTRAN IV. Several beneficial consequences result. For example, this portability feature allows easy transfer of the network component of a computer-based decision system to a new computer. It also greatly facilitates imbedding the code as a subroutine within a larger system. (This portability advantage was a major factor in achieving the computational results for network-related problems indicated in Part I.)

A final computational advantage, which will be discussed in the next section, is reduced round-off error. Taken together, the impressive array of advantages of the network solution codes makes it clear why their use in industry and government applications is rapidly increasing.

IMPLEMENTATION APPROACHES

As pointed out in the preceding sections, linear network problems are special types of LP problems and can thus be solved using any standard LP solution technique. Improvements in LP inversion and reinversion processes, data compactification, and pivot strategies have provided dramatic increases in the efficiency of primal simplex LP computer codes in recent years. In many cases, special structures such as GUB constraints (which are embodied within network problems) is detected by current LP codes. This information is then used to reduce storage requirements and to simplify operations.

In view of all this, it would seem that current LP systems could scarcely be rivalled in efficiency for solving network problems. Yet we saw in the preceding section that this is not so. The network solution systems dwarf the LP systems in their efficiency. In this section we examine some of the reasons for this, and undertake to trace their more significant implications.

Undoubtedly, the primary reason for the superiority of network codes over LP codes is the fact that the latter, with the exception of the GUB feature, are based completely on algebraic or arithmetic processing. That is, these codes maintain and update a basis inverse by manipulating numbers. Further, the representation of a variable to enter the basis is computed by specialized forms of matrix multiplication.

By contrast, the most efficient methods for solving network problems are based on replacing arithmetic operations with "logical" operations. More precisely, these solution procedures are based on viewing the problem in a graphical context (just as in the case of the network modeling ideas discussed previously).

In particular, the network codes AP-AB, ARC-II, NETG and NETSG (for assignment, transshipment, generalized, and singularly constrained generalized networks, respectively) all store the coefficient matrix and basis matrix as graphs using computer list structures.

The use of such computer list structures reduces both the amount of work needed to perform the algorithmic steps and the amount of computer memory required to store essential data. For example, network codes normally store only the cost coefficient, the upper bound, and the "to" node for each column of the coefficient matrix. (In the case of a GN problem, the multiplier is also stored.) In this way, problem data can often be resident in central memory even for large-scale problems.

Network basis matrices, like network coefficient matrices, have a graph structure. Moreover, the graph structure of a basis matrix is very special, consisting either of disjoint trees or of trees coupled with one additional arc (called *quasi-trees*). The basis of a pure network consists of a single tree and the basis of a GN problem consists of one or more trees and quasi-trees. These trees and quasi-trees can be stored and updated with remarkable efficiency by special linked list and labeling procedures that have been developed in the past few years [2, 3, 13, 17, 28].

The original problem data (compactly stored) and the basis matrix (stored via linked lists) are the only data elements kept by network codes. No basis inverse is stored. In LP systems, inverses generally require considerable amounts of storage and involve numerous (error-producing) arithmetic operations.

In network systems, the specialized labeling rules (that are designed to exploit the linked list storage structures) operate on the basis graph in a manner that obviates the use of a basis inverse.

Network Basis Traversal and Relations to Inverse Updating

The network labeling and list procedures provide a means for traversing (and modifying) relevant portions of the basis graph. Each disjoint component of the basis graph is given a "top-to-bottom" orientation. Accordingly, there are two major types of traversal required to execute the basis exchange steps, called "upward traversal" and "downward traversal."

The first, upward traversal, is associated with operations normally requiring pre-multiplication by the inverse, such as determining the representation of a variable to enter the basis. This operation is performed by traversing the unique paths from selected nodes up to their "junction" point. Simultaneously, the equivalent triangular system of basis equations is solved, in effect, by back substitution. This approach has two advantages. First, original problem data are used to compute the representation; thus, roundoff error is minimized. Second, operations involving elements of the basis representation with weights of zero, or the execution of checks to identify such elements, are eliminated since the upward traversal of the basis graph isolates the elements that receive non-zero weights.

Downward traversal is analogous to post-multiplication by the inverse. This operation is used to calculate updated dual variable values associated with the problem nodes. These values are often referred to in the literature as node potentials. At each iteration, new dual values must be computed for the nodes in a particular subtree associated with the arc leaving the basis. (The set of nodes whose potentials must be updated for GN problems has a slightly more complex characterization.) These nodes can all be encountered, and their new node potentials determined by downward traversal. For a pure network, it is only necessary to add a constant to the potential of each node in the subtree. In the case of a GN problem, each new value is computed by a simple operation which is equivalent to solving an equation in just one variable. An additional computational advantage of this procedure is that only the dual variables whose values change are examined at each iteration. Thus this operation again strictly eliminates checking or performing arithmetic operations on zero elements.

Integrated Operations

In the better network codes, the updating of the dual variables is integrated with the updating of the basis graph. The complete basis exchange step involves finding the representation of the arc to enter the basis, determining the arc to leave the basis, updating the node potentials, and restructuring (and partly reorienting) the basis graph by removing the arc leaving the basis and inserting the arc entering the basis. This last step, which is equivalent to updating the basis inverse, is accomplished simply by changing a few pointers in the list structures: no arithmetic operations are involved and, consequently, no round-off error introduced. The integration of this basis update with the dual update, by which both processes are carried out simultaneously, is made possible by the specialized labeling procedures [2].

These advantageous features of network systems have motivated researchers to develop extensions for solving LP problems with embedded networks. The NETSG code is an instance of one such extension. The next section briefly describes the fundamental ideas underlying such extensions and indicates their potential value. On the basis of our past experience, we judge these extensions to represent the wave of the future--the very near future--in mathematical programming.

EXPLOITING EMBEDDED NETWORK STRUCTURE

Powerful procedures recently proposed for exploiting embedded network structure are based upon partitioning the coefficient matrix of an LP problem into network and non-network components. Such a partitioning scheme arranges the rows so that each column has at most two non-zero entries in the first m rows out of a total of m + q rows. By reference to this partitioning, a basis compactification procedure is employed that induces an identical row partitioning on the basis matrix B so that B can be expressed in the form

$$B = \begin{bmatrix} G_m & E_q \\ A_m & A_q \end{bmatrix}$$

The submatrix G_m is an m x m basis for the underlying network (or networks) composing the first m row of the LP problem. The basis inverse B^{-1} may be stated relative to this same partition as:

$$B^{-1} = \begin{bmatrix} (G_m^{-1}+G_m^{-1}E_qQ^{-1}A_mG_m^{-1}) & (-G_m^{-1}E_qQ^{-1}) \\ -Q^{-1}A_mG_m^{-1} & Q^{-1} \end{bmatrix}$$

where $Q = A_q - A_mG_m^{-1}E_q$.

The motivation for this partitioning is to factor out the matrix G_m. By its connection with the network, G_m may be stored as a graph and any operations involving G_m^{-1} may be performed by the special labeling and basis traversal techniques discussed in the preceding section. This means that G_m^{-1} need not be explicitly generated and the full basis inverse B^{-1} can be determined simply by referring to the partitioned components of the basis B and the q x q matrix Q^{-1}. Thus, Q^{-1} may be viewed as the *working basis inverse* for such problems. Since the row (and column) dimension of Q^{-1} is equal to the number of non-network constraints of the LP problem, the algorithmic steps of the simplex method can be executed much more efficiently and with considerably less demand on computer memory by merging the network updating of G_m^{-1} with the standard updating of Q^{-1}. The benefits previously discussed for dealing with network basis updating thereby carry over to the partitioned LP problem with only the residual portion of the basis associated with Q^{-1} being subject to the slower customary process with its greater attendant susceptibility to round-off error and numerical inaccuracy. In addition, since the ordering of the rows to achieve the partition need be done only once (and often will automatically be accomplished by the initial formulation), the augmentation of commercial LP systems with such a "special order network" (*SON*) feature would constitute a noteworthy and beneficial enhancement.

The development of the SON feature will in our opinion produce the next major computational advance in large-scale linear programming. In fact, it should have a more profound effect than the development of generalized upper bounding (GUB). This belief is based on the following:

1) GUB is a specialization of these procedures. This can be seen simply by observing that the GUB constraints can be scaled and summed to form another constraint which, when appended to the GUB constraints, yields a network.

2) SON extends GUB in several important ways. For example, it eliminates the non-overlapping variable requirement of GUB in an efficient manner. Other attempts to accomplish this by "generalized GUB" procedures do not share the computational power of network techniques.

3) Computational experience with the SON feature in prototype applications involving both a simple form of the problem, solved completely by SON, and a fully general form of the problem, solved partly by SON, already demonstrate that substantial computational savings are possible. In particular, the NETSG code, previously reported to be 25 times faster than APEX-III (and with substantially better memory requirements), is a direct application of SON to the situation of a single arbitrary linear constraint, where the network portion is a generalized network with arbitrary multipliers.

At the opposite end of the spectrum, we have implemented a "first pass" execution of the SON feature for a major automobile manufacturer engaged in solving more general LP problems with large imbedded network components. The firm was solving the LP problems of this application on an IBM 370/145 using MPSX, incurring computer run times in excess of 30 minutes per problem. We superimposed a network system on the MPSX system in a manner designed to operate on the network portion of the problem to produce an advanced starting basis for MPSX. This "first pass" application by itself reduced the total solution time from over 30 minutes to 10 minutes.

SON and Integer Programming

There is another important application of the SON feature that extends beyond its use in solving LP problems. This application has to do with solving pure and mixed integer programming (IP) problems. Current research has shown that the best solution approach is to use formulations which keep the number of integer variables as small as possible and which yield strong LP relaxations, rather than minimizing the number of constraints. Fortunately the IP problem manipulation schemes for obtaining stronger LP relaxations often induce network structure. This result is largely unrecognized presently by many IP practitioners and even by researchers. To illustrate this statement, consider the constraint

$$x_1 + x_2 + x_3 - 3x_4 \leq 0$$

where x_1, x_2, x_3, x_4 are 0-1 variables. It is well known that replacing this constraint by the constraints:

$$\begin{aligned} s_1 + x_1 &= x_4 \\ s_2 + x_2 &= x_4 \\ s_3 + x_3 &= x_4 \end{aligned}$$

yields a stronger LP relaxation. Note however that by performing elementary row operations this latter set of constraints is equivalent to:

$$\begin{array}{llllll} 0 = x_1 - x_2 & & +s_1 - s_2 & & & \\ 0 = & x_2 - x_3 & & +s_2 - s_3 & & \\ 0 = & x_3 & & & s_3 - x_4 & \end{array}$$

Subtracting these constraints produces $0 = -x_1 - s_1 + x_4$. Appending this equation to the others yields a network.

By combining the IP problem manipualtion approach with the SON feature the working basis inverse of the LP problem would be reduced by the number of constraints that would otherwise be added to accommodate relationships of the form indicated. Thus the addition of the SON feature to commercial LP systems would substantially overcome the problem expansion concern of many practitioners when using stronger IP/LP formulations.

The above example is only one of many important uses of the SON feature in integer programming. Almost every practical IP application - fixed charge, location allocation, project selection, capacity expansion, and set-up scheduling problems have substantial network substructures.

In sum, the SON feature would evidently constitute an important computational advance for large-scale LP and IP. Presently, however, no LP or IP system has this feature. The full extent of the advantages to be derived from the SON feature needs to be established by extensive testing in diverse applications. The potential of the approach, in our opinion, strongly merits such a thorough implementation and analysis.

REFERENCES

1. Analysis, Research, and Computation, Inc., "Development and Computational Testing on a Capacitated Primal Simplex Transshipment Code," ARC Technical Research Report, P.O. Box 4067, Austin, Texas 78765.

2. R. Barr, F. Glover, and D. Klingman, "Enhancements of Spanning Tree Labeling Procedures for Network Optimization," Research Report CCS 262, Center for Cybernetic Studies, University of Texas at Austin, 1976.

3. R. Barr, F. Glover, and D. Klingman, "The Alternating Basis Algorithm for Assignment Problems," Research Report CCS 263, Center for Cybernetic Studies, University of Texas at Austin, 1977.

4. G. Bhaumik, *Optimum Operating Policies of a Water Distribution System with Losses,* Unpublished Dissertation, University of Texas at Austin, August, 1973.

5. G. Bradley, G. Brown, and G. Graves, "Design and Implementation of Large Scale Primal Transshipment Algorithms," Technical Report NPS55BZBW76091, Naval Postgraduate School, Monterey, California, 1976.

6. A. Charnes and W. Cooper, *Management Models and Industrial Applications of Linear Programming*, Vols. I and II, Wiley, New York, 1961.

7. R. Crum, "Cash Management in the Multinational Firm: A Constrained Generalized Network Approach," Working Paper, The University of Florida, Gainesville, Florida, 1976.

8. G. Dantzig, *Linear Programming and Extensions,* Princeton University Press, Princeton, New Jersey, 1963.

9. J. Evans, "A Single Commodity Network Model for a Class of Multicommodity Dynamic Planning Problems," Working Paper, University of Cincinnati, 1975.

10. G. Gilliam and J. Turner, "A Profile Analysis Network Model to Reduce the Size of Microdata Files," Working Paper, Office of Tax Analysis, Office of the Secretary of the Treasury, Washington, D.C., 1974.

11. J. Gilsinn and C. Witzgall, "A Performance Comparison of Labeling Algorithms for Calculating Shortest Path Trees," NBS Technical Note 772, U.S. Department of Commerce, 1973.

12. F. Glover, J. Hultz, D. Klingman, and J. Stutz, "A New Computer-Based Planning Tool," Research Report CCS 258, Center for Cybernetic Studies, University of Texas at Austin, 1976.

13. F. Glover, D. Karney, and D. Klingman, "The Augmented Predecessor Index Method for Locating Stepping Stone Paths and Assigning Dual Prices in Distribution Problems," *Transportation Science*, 6, 2 (1972), 171-179.

14. F. Glover, D. Karney, and D. Klingman, "Implementation and Computational Study on Start Procedures and Basis Change Criteria for a Primal Network Code," *Networks*, 4, 3 (1974), 191-212.

15. F. Glover and D. Klingman, "Network Application in Industry and Government," Research Report CCS 247, Center for Cybernetic Studies, University of Texas at Austin, 1975.

16. F. Glover, D. Klingman, and C. McMillan, "The NETFORM Concept," Research Report CCS 281, Center for Cybernetic Studies, University of Texas at Austin, 1977.

17. F. Glover, D. Klingman, and J. Stutz, "The Augmented Threaded Index Method for Network Optimization," *INFOR*, 12, 3 (1974), 293-298.

18. F. Glover and J. Mulvey, "Equivalence of the 0-1 Integer Programming Problem to Discrete Generalized and Pure Networks," MSRS 75-19, University of Colorado, Boulder, Colorado, 1975.

19. R. Helgason, J. Kennington, and H. Lall, "Primal Simplex Network Codes: State-of-the-Art Implementation Technology," Technical Report IEOR 76014, Department of Industrial Engineering and Operations Research, Southern Methodist University, Dallas, Texas, 1976.

20. J. Hultz and D. Klingman, "Solving Singularly Constrained Generalized Network Problems," Research Report CCS 256, Center for Cybernetic Studies, University of Texas at Austin, 1976.

21. J. Hultz and D. Klingman, "Solving Constrained Generalized Network Problems," Research Report CCS 257, Center for Cybernetic Studies, University of Texas at Austin, 1976.

22. P. Kazemersky, *A Computer Code for Refueling and Energy Scheduling Containing an Evaluator of Nuclear Decisions for Operation*, Unpublished Dissertation, Ohio State University, 1974.

23. Y. Kim, "An Optimal Computational Approach to the Analysis of a Generalized Network of Copper Refining Process," Presented at the Joint ORSA/TIMS/AIIE Conference, Atlantic City, New Jersey, 1972.

24. D. Klingman and R. Russell, "On Solving Constrained Transportation Problems," *Operations Research*, 23, 1 (1975), 91-107.

25. M. Collins, L. Cooper, and J. Kennington, "Solving the Pipe Network Analysis Problem Using Optimization Techniques," Technical Report IEOR 76008, Southern Methodist University, 1976.

26. "Out-of-Kilter Network Routine," SHARE Distribution 3536, SHARE Distribution Agency, Hawthorne, New York, 1967.

27. T. Ross and R. Soland, "A Branch-and-Bound Algorithm for the Generalized Assignment Problem," *Mathematical Programming*, 9 (1975), 91-103.

28. V. Srinivasan and G. Thompson, "Accelerated Algorithms for Labeling and Relabeling of Trees with Applications for Distribution Problems," *JACM*, 19, 4 (1972), 712-726.

29. H. Taha, *Operations Research*, The Macmillan Company, New York, 1971.

30. L. Tavis, R. Crum, and D. Klingman, "Implementation of Large-Scale Financial Planning Models: Solution Efficient Transformations," Research Report CCS 267, Center for Cybernetic Studies, University of Texas at Austin, 1976.

31. H. Wagner, *Principles of Operations Research*, Prentice-Hall, Englewood Cliffs, New Jersey, 1969.

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Center for Cybernetic Studies The University of Texas	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP

3. REPORT TITLE

(6) Improved Computer-Based Planning Techniques.

4. DESCRIPTIVE NOTES (Type of report and inclusive dates) (9) Research rept.,

5. AUTHOR(S) (First name, middle initial, last name)

(10) Fred/Glover, John Hultz Darwin/Klingman

6. REPORT DATE (11) Feb~~ruary 1~~977	7a. TOTAL NO. OF PAGES 49	7b. NO. OF REFS 31
8a. CONTRACT OR GRANT NO. N00014-75-C-0616;~~0569, and~~ b. PROJECT NO. N00014-76-C-0383 (15) NR 047-021 c. d. (12) 53p.	9a. ORIGINATOR'S REPORT NUMBER(S) Center for Cybernetic Studies Research Report CCS-283	
	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) (14)	

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Office of Naval Research (Code 434) Washington, D. C.

13. ABSTRACT

Management Science has responded recently to the needs of practitioners by contributing two new technologies. These are network computer implementation technology and NETFORM (network formulation) technology. In this paper we show how these new technologies have been used to model and solve real-world problems. In addition, we attempt to give the practitioner insights into how these important policy evaluation tools may be applied to his unique management problems.

DD FORM 1 NOV 65 1473 (PAGE 1)

406 197

Unclassified

Unclassified
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Linear Programming Networks Graphs Transportation Transshipment Modeling						

DD FORM 1 NOV 65 1473 (BACK)
S/N 0102-014-6800

Unclassified
Security Classification

A-31409